



DATABASE Expert

A hush descends on the column this month as **Kay Ewbank** enters the spreadsheet library to explain how to use Excel functions in Access, and looks at a product that helps you keep an audit trail of your data



SQL SEQUEL

EMS has released a new version of its low-cost tool for building SQL queries graphically. SQL Query 2005 can be used to create queries for MySQL, PostgreSQL, Microsoft SQL Server, Interbase/Firebird and IBM DB2 without writing SQL. The graphical interface makes it easy to connect to databases, select tables and fields for a query, set the selection criteria and so on. The new version adds a column summary facility and a quick data search option. A 30-day trial can be downloaded from www.sqlmanager.net/downloads. Registration costs \$65 (around £38).

PHP SPEED BOOST

PHP, the main open-source language used for web database development, has been updated. Version 5.1.0 of the scripting language is being described as the most significant upgrade since the release of version 5.0 in 2004. The main improvements are performance benefits, though database developers will also be pleased with an extension aimed at simplifying interaction with databases.

It is estimated that PHP is now used on more than 22 million domains, and that 40 per cent of all web applications run PHP. Major companies that use it include Yahoo!, Lufthansa and T-Online. See www.zend.com for more details.

Q I need to carry out some statistical calculations on data entered in Access. Unfortunately, Access doesn't have the functions that I need. Excel, however, does. Can I call an Excel function in Access queries? *Billy Carpenter*

A You can use Excel functions, but it involves writing some VBA code. First you need to add a reference to whichever Excel library you have on your PC. Open the code editor in Access (by selecting the Build Event property of a command button, for example) and in the Tools menu, choose References. From the list of available references, select the one for Microsoft Excel. The exact version of the library you'll see will depend on the version of Excel you're using. Excel 2003 has a library called Microsoft Excel 11 Object Library. Excel 2000 shows up as Excel 9. They all offer similar functions, so as long as you have one of them available, the steps that follow should work.

Once you've added the reference to the object library, you can refer to Excel functions in your Access code. To do so, first set up a reference to an Excel Application object using:

```
Dim xl As New Excel.Application
```

Then call the function you want to use. If you wanted to use the Max function, you'd use:

```
mymax = xl.WorksheetFunction.Max(4, 6, 3, 5)
```

In most cases you can use a shorter form of this:

```
mymax = xl.Max(4, 6, 3, 5)
```

However, if you include the reference to WorksheetFunction, the Access code editor will look up what functions are available and autocomplete

them as you type, as well as reminding you what needs to go in the brackets.

While the syntax above works with 'normal' Excel functions, you need to do a bit more work if you want to use functions from any of the Excel add-ins. For example, if you wanted to use a function from the analysis add-in, you'd need to open it as part of your code. So you might have:

```
Dim xl As New Excel.Application
xl.workbooks.Open (xl.Application.Libr
arypath & "\Analysis\atpvbaen.xla")
xl.workbooks("atpvbaen.xla").RunAuto
Macros (xlAutoOpen)
```

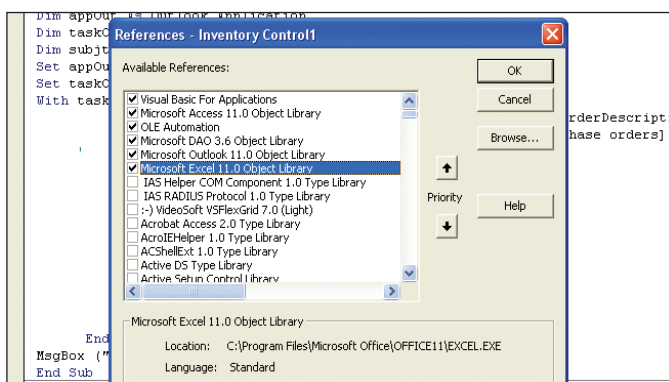
The second line opens the analysis add-in, the librarypath property telling Access where to find it. The third line runs the AutoOpen macro in the add-in to make the functions the add-in contains available to your code. After this, you can use the functions as you would those built into Excel.

The final choice you need to make is how to handle the instance of Excel you've opened with the Dim command. There are two ways to go about this. You can open a new instance of Excel every time you want to use a function, remembering to close it again when you've used the function, or you can open it once and leave it open until you've finished using it. The first method has the advantage that you won't accidentally leave an Excel instance running. It is slower, though, because each time you call the function you have to start Excel.

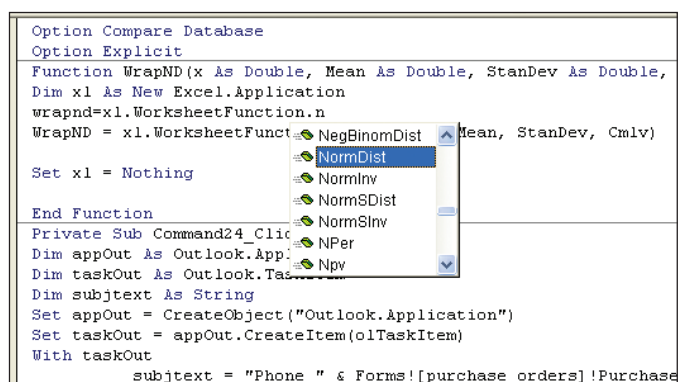
Whichever option you choose, close the instance of Excel by including the following lines:

```
xl.Quit
Set xl = Nothing
```

Now you've seen how to use Excel functions in Access code, the next task is to use the functions in your Access queries. To do this, you create a



Check you've added a reference to the Excel library using the Tools, References menu before attempting to use Excel functions in Access



So long as the reference is correctly added, Access will autocomplete the Excel functions for you



REVIEW DATABASE UTILITY

INNOVARTIS DB Ghost

RATING ★★★★★

PRICE £199 inc VAT

DETAILS www.innovartis.co.uk

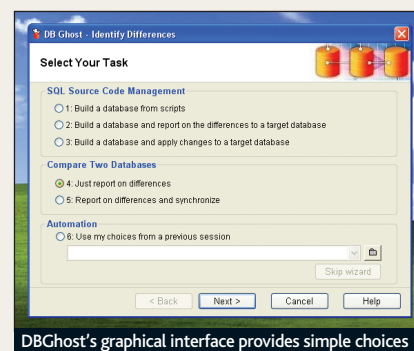
The first version of a database usually works well. You've carefully planned and created all the structures so they do just what you want. Things tend to go wrong as a database grows up. Inevitably you need to make changes to the original design, and on larger databases several people may need to make changes. This introduces problems such as ensuring everyone is using the same schema. You may have a production copy and a development copy of a database, or a live version and a report version. How can you be sure that all the versions are structurally identical? Many companies also now require an audit trail showing how structures were changed, when and by whom.

One answer is to use a utility that compares databases and reports the differences. A smarter

solution is offered by DB Ghost. This provides a set of components you can use to manage your database structures, and make changes to structures in a way that can be audited. You connect DB Ghost to your source code control system, and it extracts the definitions of your database objects and look-up data from there. This information is stored as verified code. You can then use this code to compare and synchronise the source database with a target database. If there are any differences between the two, DB Ghost generates a change script. You can use the scripts generated by DB Ghost to create a new database with the same structure.

Whenever a version of your database schema is scripted in DB Ghost, it can be stored in the repository. This makes it easy to go back to an earlier schema. You can also use the archive as an audit trail.

While not everyone will have multiple versions of databases on different servers, another element of DB Ghost will prove more universally useful. Once you've set up a schema, you can use it to verify any stored procedure you're working on. You pass the stored procedure to DB Ghost, and your code is verified to ensure that the references within it match your database. You can also have DB Ghost check for dependency errors.



DBGhost's graphical interface provides simple choices

The new version of DB Ghost has been broken into components, so you can buy just the facility to use a script to create a new database or just the schema comparison element. The Professional version provides all four components. You can also download a free trial version.

✓ Easy to use; gives good control over schema changes

✗ Specific to SQL Server

wrapper function in Access. This is a piece of Access code that is declared as a Function so you can use it in queries, and that takes the same arguments as the Excel function it wraps. The wrapper function calls the Excel function, passing it the arguments. A typical wrapper function looks like this:

```
Function WrapND(x As Double, Mean As Double, StanDev As Double, Cmltv As Boolean) As Double
Dim xl As New Excel.Application
WrapND = xl.WorksheetFunction.NormDist(x, Mean, StanDev, Cmltv)
xl.Quit
Set xl = Nothing
End Function
```

In your query, you can use the wrapper function WrapND to access the function NormDist like this:

```
CalculatedField: WrapND([FieldName1], 8, [FieldName2], true)
```

This works out the normal distribution for fieldname1 based on a standard deviation stored in fieldname2, with a mean of 8. All statistical functions can be wrapped in the same way, making the full range of Excel calculations available in Access.

Q I'm creating a report in Access based on a query, QryInStock. The report also includes a text box whose source is a DSUM expression:

```
=DSUM("StockLevel", "QryInStock", "ProductType=" & [ProdType])
```

As I understand it, this expression will sum all the stock levels of the products returned by the query QryInStock whose ProductType is ProdType. I use this expression to summarise predicted stock levels, as the query QryInStock lists my stock under various circumstances.

The problem I have is that when I run the report, the text box shows #Error. Why is this happening?

Paul Smithson

A DSUM allows you to sum data in records from a source you specify, matching criteria you specify. The syntax is:

```
DSum(expression to sum, record source, criteria as string)
```

Your basic syntax is correct, but there are lots of other things that can go wrong with DSUM, and Access can be a bit cryptic with its error messages. For example, you'll get an output of #Error if you try to use DSUM to calculate the sum of a control on your report that is itself calculated. So if you had a calculated field such as DiscountedPrice, which is worked out from UnitPrice and Discount fields, and you tried to calculate the DSUM for that field, you'd get an error. The way around this problem would be to include the calculation in the DSUM expression. However, this does not seem to be what's happening here.

The most likely cause of your problem is the criteria string. If you had a group where the ProdType control was empty, you'd be asking to DSUM a value where the selection criteria was:

```
ProductType=
```

This is nonsensical, hence the #Error answer.

You could check whether this is likely to be the problem by changing your criterion to have a static test value. So you might change it to read:

```
=DSUM("StockLevel", "QryInStock", "ProductType=2")
```

This assumes 2 is a valid product type. If that worked, you would know that the problem lies with the values that are being passed in via ProdType.

You could get around the problem by using the NZ function to look for a null value in ProdType, and replace it with something else. For instance:

```
=DSUM("StockLevel", "QryInStock", "ProductType=" & NZ([ProdType], 0))
```

This would mean that if the ProdType wasn't completed, DSUM would get a selection criterion of:

```
ProdType=0
```

As long as you had no valid ProdType of 0, DSUM wouldn't select any records, and would return a zero stock level, which is far better than an error.

You might also want to contemplate whether DSUM is the best function to use in your situation. The DSUM function is designed to calculate the sum for any table or query that you refer to in its criteria. It is useful when you need to sum values from a field that is not in the record source for your form or report – that is, an 'external' table or query.

In your case, you may be able to use the much simpler SUM function, which works only on fields in the current recordset. SUM will work on the controls and calculations in the Group of a Totals query, the groups of forms or reports or the entire recordset if you put it in the report header or footer. So if your report shows the data from QryInStock, you can use SUM just as easily as you could DSUM. SUM would be calculated much more rapidly than DSUM, particularly if you have a relatively small recordset for your report, as SUM would consider only those records. **CS**

CONTACT

KAY EWBank

Kay is Computer Shopper's database expert and has worked as a consultant and author for 20 years

kay@computershopper.co.uk